Machine Learning-assisted word feature finding in the Bible

1 Overview

The advent of the information technology and the rapid development and popularization of Internet technology have greatly accelerated the speed of information dissemination, and the amount of data has surged, bringing with them all the distress of "we are overwhelmed by information, but lack knowledge". There is an urgent need for society to mine the textual data in the vast amount of web data, to understand them, to extract important patterns and trends from them, and to understand what these data "say".

Text mining is data mining with text data as the mining object. Content of text data needs to be structurally analyzed, summarized, classified, clustered, correlation analysis, distribution analysis, and trend prediction for text mining. In this assignment, the four steps of text mining routine will be performed: feature representation, feature dimensionality reduction, pattern mining, and pattern evaluation. I use an unsupervised machine learning approach, word2vec for feature representation and build a neural network (NN) model to analyze the text in a book. My choice of book is the Bible. I am an atheist and completely unfamiliar with biblical culture and have never read the Bible. I chose the Bible out of curiosity about its patterns in text.

Machine learning (ML) (Mitchell, 2007) is a branch of AI that focuses on the exploitation of data and algorithms to imitate human learning and progressively improve its accuracy. In this task, I will use ML algorithm build a natural language model, adopt the model to review the Bible text, and try to find the text patterns/features in the Bible.

Traditional natural language processing (NLP)(Chowdhary and Chowdhary, 2020) approaches, such as Bag-of-Words (BOW) model use one-hot encoding (each component $x \in \{0, 1\}$ for every dimension) to code the words and vectorize the text. Instead of considering syntax, or even word order, but preserving polysemy, a text is expressed as a bag of its words (multiset). There is no free lunch here. The disadvantages of BOW are the loss of word order information (in other words, BOW cannot capture the semantic relationship between words), and the data in the vector space are too sparse. Alternatively, word2vec method vectorize the words in a way that, each component x can take values in the domain of natural numbers ($x \in \mathbf{N}$). By doing so, the dimension of the vector space for the text could be greatly reduced. The question is how do we determine the value in each direction? Word2Vec(Church, 2017) gives us a solution. One

commonly used algorithm is skip-gram. Briefly speaking, the skip-gram first use one-hot vector to code the text and use neural network to train the model. This NN has one hidden layer, in which the number of the neurons are smaller than the input layer. The model is trained such that the output layer output the possibility which is close to the training data sets. The words then can be vectorized through this hidden layer, and the dimension is reduced. Interestingly, the similar words are getting closer in space, and we can also get some information from the vector difference, e.g. King-Queen=man-women.

2 Text data

In the following steps, I will use skip-gram to train the NN, and vectorize the word in the Bible. The version of the Bible I have chosen is the King James Version (KJV), which is an English translation of the Bible published in 1611 under the presidency of England's King James I. From the middle of the 17th century to the early 20th century, this version of the translation was commonly acceptable as the standard English Bible. The language style of this edition may differ from modern English style. Included in the 80 books of the King James Version are thirty-nine books of the Old Testament (OT), fourteen books of the Apocrypha, and twenty-seven books of the New Testament (NT). The KJV has been described as one of the most important books in English language culture, even for English literature and thus can serve as a typical corpus for study.

In preparation for analyzing the data, several pre-processing steps were performed, including tokenization, transformation of all text to lowercase, and removal of punctuation and stop words. No stemming was performed here.

The data I download is in the .text format with UTF8 encoding. The data is read by the file.read() function. With the help of the nltk tools. The data is tokenized. Here the first 2850 words are Copyright description text and table of contents, and hence they are excluded from the study.
In the next step, the punctuations are removed. In the KJV, each sentence is initiated by an index like 1:10. These indices are also removed from the sentence. These “cleaned” sentences are separated. The total number of the sentences in KJV is 29926.

3 Method

3.1 NLP sentiment analysis

I adopt sentiment intensity analyzer from nltk tools to make the sentiment analysis. The vader lexicon is used to evaluate the sentimentality of sentences. For each sentence, there is a neg score, a neu score and a pos score, and they are weighed summed to a compound score. The compound scores range from -1 to 1, and 0 stand for neutral. Most of the sentences are neutral. The most positive sentence is from Genesis 49:22 "Even by the God of thy father, who shall help thee; and by the Almighty, who shall bless thee with blessings of heaven above, blessings of the deep that lieth under, blessings of the breasts, and of the womb" with the score of 0.9657. The most negative sentence is from Numbers 35:20 "But if he thrust him of hatred, or hurl at him by laying of wait, that he dies" with the score of -0.9209. (See Figure 1 in the appendix)

3.2 Word2Vec Modelling

In this part, I vectorize all the words appeared in KJV by the Word2Vec model. All the words with a total frequency lower than 3 were ignored by me. The word vector (hidden layer) has a dimension of 100. Based on this model, one can list all the similar words to the word I am interested. For example, I can list ten closest words to God. Interestingly, father is one of the most similar words, with the similarity factor of 0.69. Saviour and Redeemer rank first and second on the list, which I believe that the theological scholars would agree. I also calculate the ten similar words to Jesus, i.e., Christ John, Nazareth, Lord, Paul, baptism, Esaias, gospel, Peter, resurrection. The apostle Paul John and Peter seems to be the closest ones to Jesus. (see Figure 2&3 in the appendix)

4 Evaluation and findings

I stored the model and save it into a TSV format. This format is suitable for data visualization by tensorflow projector.[1] These vectorized data are therefore PCA analyzed. I can see the following findings from our analysis. One interesting phenomenon is that the word God seem to be located at the edge of the data space. Another finding is that most sentences in the Bible are neutral. (see Figure 4 in the appendix)

Finally, the frequency distribution of different words in Bible KJV is not significantly different from the frequency distribution of modern English words. One topic that can be studied is to take Shakespeare's play collections as the research object, and compare the distribution of words in the play collections with those in modern English and Bible KJV, so as to study the evolution of modern English.

---

[1] https://projector.tensorflow.org/

References:

CHOWDHARY, K. R. & CHOWDHARY, K. R. (2020), "Natural language processing", *Fundamentals of artificial intelligence,* 603-649.

CHURCH, K. W. (2017), "Word2Vec", *Natural Language Engineering,* Vol. 23 No. 1, pp. 155-162.

MITCHELL, T. M. (2007), Machine learning, McGraw-hill New York.

Appendix A: Figures

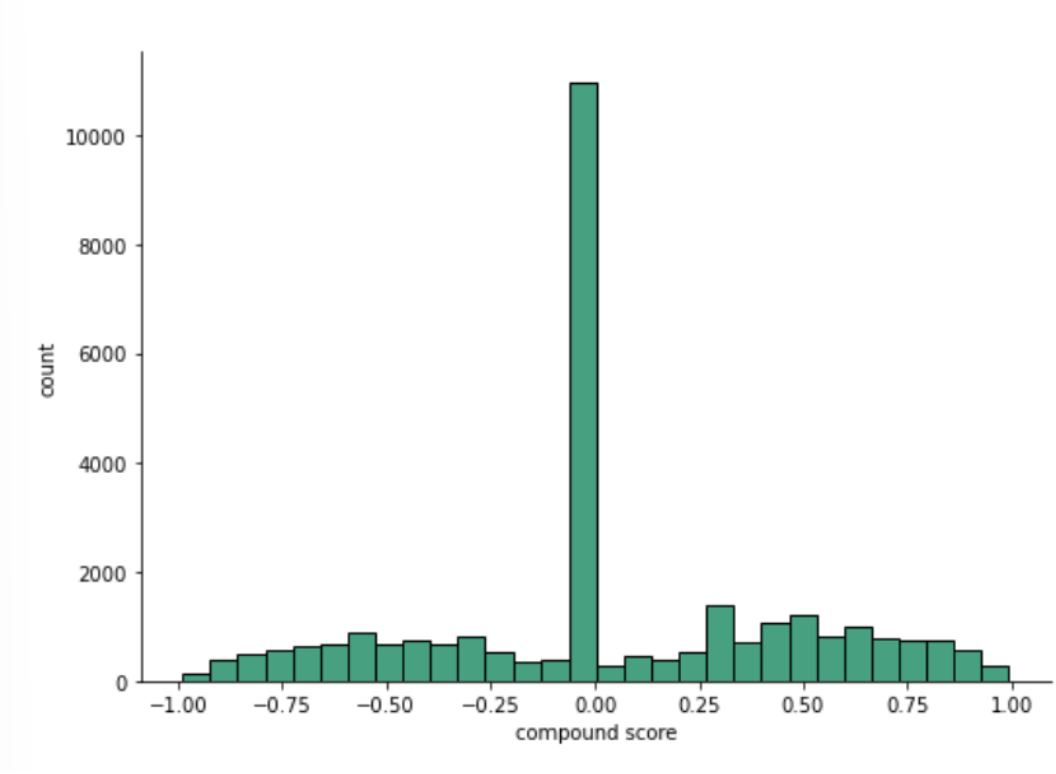

Figure 1 compound score for all the sentence in KJV

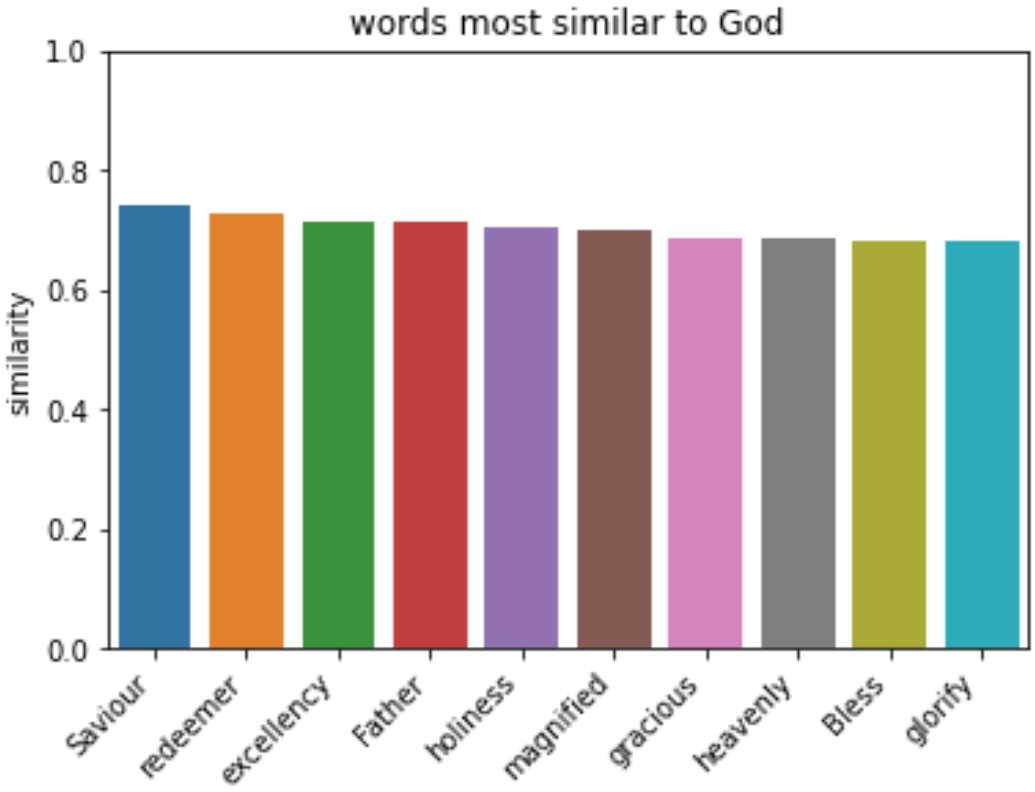

Figure 2 words that are most similar to God

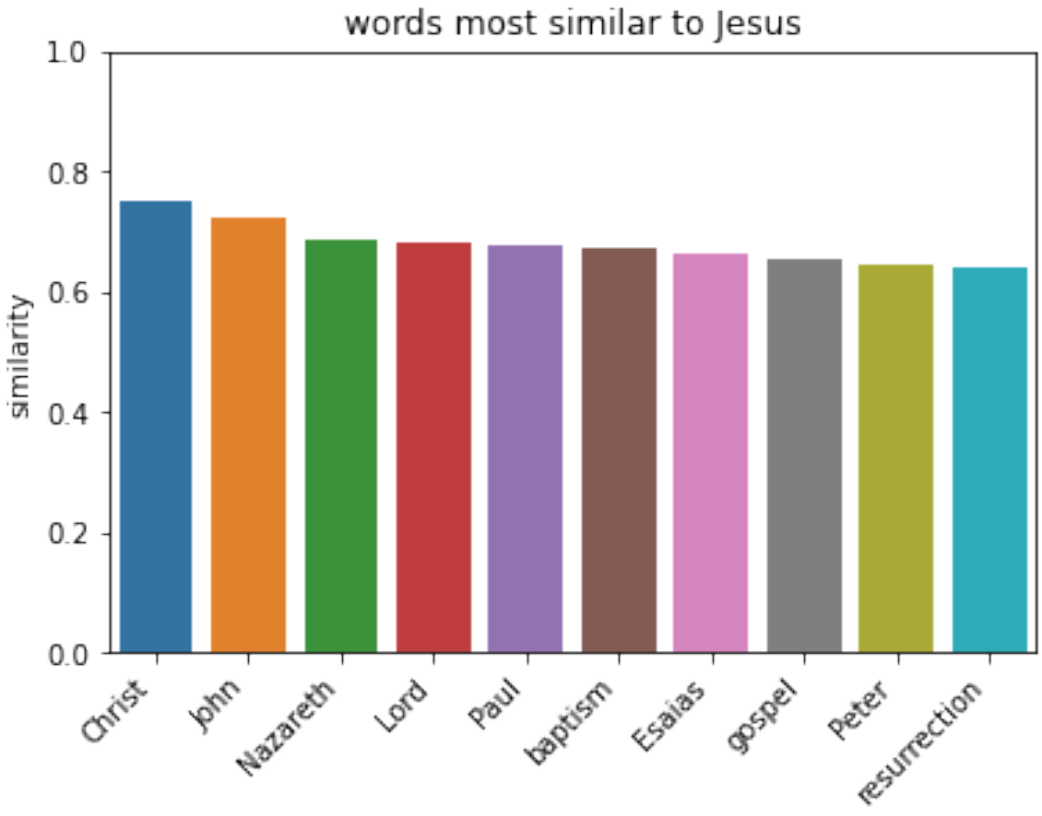

Figure 3 words that are most similar to Jesus

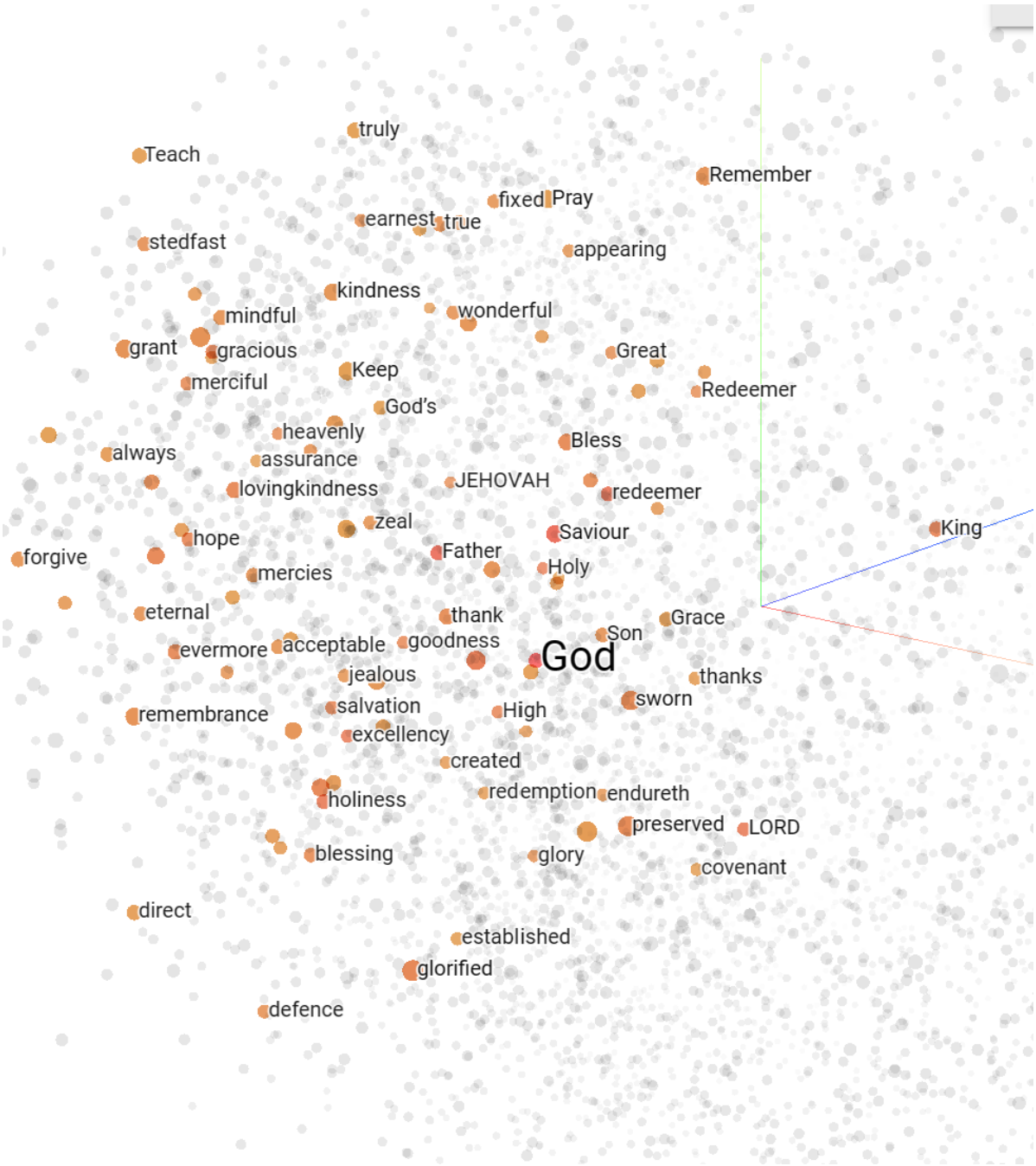

Figure 4 PCA analyze to the data